**REACT & LARAVEL**

# Bài 1. Cài đặt môi trường

**A. Cài đặt nodejs**

Để cài đặt Node.js trên Windows, bạn có thể làm theo các bước sau:

**Bước 1:** Tải xuống Node.js

Truy cập trang web chính thức của Node.js tại đây: https://nodejs.org/

Tại trang chính, bạn sẽ thấy hai phiên bản để tải xuống:

LTS (Long Term Support): Đây là phiên bản ổn định và được khuyến nghị cho hầu hết mọi người. Nó được hỗ trợ trong thời gian dài và thích hợp cho các dự án lớn.

Current: Đây là phiên bản mới nhất với các tính năng và cập nhật mới nhất. Thích hợp cho việc thử nghiệm và phát triển ứng dụng, nhưng không được hỗ trợ lâu dài.

Chọn phiên bản mà bạn muốn cài đặt và tải xuống bằng cách nhấn vào nút tải về tương ứng. Hãy chọn phiên bản mà bạn cảm thấy phù hợp nhất với dự án của mình.

**Bước 2:** Cài đặt Node.js trên Windows

Khi tải xong tệp cài đặt, mở tệp đó bằng cách nhấp đúp vào nó.

Trong cửa sổ cài đặt, bạn có thể chọn ngôn ngữ và nhấn "Next" để tiếp tục.

Chấp nhận các điều khoản bằng cách đánh dấu ô "I accept the terms in the License Agreement" và nhấn "Next".

Chọn nơi bạn muốn cài đặt Node.js. Mặc định, nó sẽ được cài đặt trong thư mục "C:\Program Files\nodejs". Sau đó, nhấn "Next".

Chọn các tính năng bạn muốn cài đặt. Thông thường, bạn có thể giữ mặc định và nhấn "Next".

Đợi cho quá trình cài đặt hoàn thành.

Sau khi cài đặt xong, bạn có thể kiểm tra phiên bản Node.js và npm bằng cách mở Command Prompt hoặc PowerShell và chạy các lệnh sau:

Kiểm tra phiên bản Node.js:

node -v

Kiểm tra phiên bản npm:

npm -v

Bây giờ bạn đã cài đặt thành công Node.js trên Windows và có thể bắt đầu sử dụng nó để phát triển ứng dụng Node.js trên máy tính của bạn.

**B. Laragon**

Laragon là một môi trường phát triển web trên nền tảng Windows dễ sử dụng để phát triển và quản lý ứng dụng web dựa trên PHP và MySQL. Dưới đây là hướng dẫn cài đặt Laragon trên Windows:

Bước 1: Tải xuống Laragon

Truy cập trang web chính thức của Laragon tại https://laragon.org/download/.

Tải xuống phiên bản Laragon phù hợp với hệ điều hành Windows của bạn. Laragon hỗ trợ cả phiên bản 32-bit và 64-bit, vui lòng tải phiên bản phù hợp.

Bước 2: Cài đặt Laragon

Sau khi tải xuống, mở tệp tải về (có thể là một tệp .exe) và chạy nó.

Trong quá trình cài đặt, bạn sẽ thấy một số tùy chọn sau:

Installation Folder: Chọn thư mục mà bạn muốn cài đặt Laragon. Mặc định, nó sẽ được cài đặt trong thư mục "C:\laragon".

Components: Bạn có thể chọn các thành phần bổ sung như Apache, MySQL, PHP, Node.js, và Redis tùy theo nhu cầu của dự án của bạn. Hãy chắc chắn rằng ít nhất bạn đã chọn "Apache", "MySQL", và "PHP".

Virtual Hosts: Bạn có thể tạo một số máy chủ ảo để phát triển các dự án riêng biệt. Thường thì bạn có thể giữ mặc định và sau đó cấu hình thêm sau này.

Nhấn "Next" và đợi cho quá trình cài đặt hoàn thành.

Sau khi cài đặt xong, bạn có thể bật Laragon bằng cách mở ứng dụng từ menu Start hoặc bằng cách tìm Laragon trong thư mục cài đặt và mở nó.

Bước 3: Sử dụng Laragon

Khi Laragon đã được cài đặt và chạy, bạn có thể thực hiện các tác vụ sau:

Bật và tắt máy chủ Apache và MySQL bằng cách nhấn vào biểu tượng Laragon trong khay hệ thống và chọn tương ứng.

Tạo các máy chủ ảo và quản lý các dự án web của bạn.

Cấu hình PHP, Apache, MySQL, và các thành phần khác.

Sử dụng phpMyAdmin để quản lý cơ sở dữ liệu MySQL.

**C. Cài đặt composer**

Để cài đặt Composer trên Windows, bạn có thể thực hiện các bước sau:

Bước 1: Tải Composer

Truy cập trang web chính thức của Composer tại https://getcomposer.org/download/.

Tại trang này, bạn sẽ thấy liên kết để tải Composer-Setup.exe. Nhấp vào liên kết đó để tải tệp cài đặt.

Bước 2: Cài đặt Composer

Mở tệp Composer-Setup.exe mà bạn đã tải về.

Quá trình cài đặt sẽ hiển thị. Trong quá trình cài đặt, bạn có thể thay đổi thư mục mà Composer sẽ được cài đặt vào. Mặc định, Composer sẽ được cài đặt trong thư mục C:\ProgramData\ComposerSetup\bin. Bạn có thể giữ mặc định hoặc chọn một thư mục khác nếu cần.

Chọn "Next" để tiếp tục.

Tiếp theo, bạn sẽ thấy một tùy chọn để chọn cài đặt PHP. Composer yêu cầu một phiên bản PHP để hoạt động. Nếu bạn đã cài đặt PHP trước đó, Composer sẽ tự động phát hiện và sử dụng nó. Nếu không, bạn có thể chọn cài đặt PHP bằng cách chọn "Install PHP" và sau đó chọn phiên bản PHP mà bạn muốn sử dụng.

Nhấn "Install" để bắt đầu quá trình cài đặt. Chờ đợi cho đến khi cài đặt hoàn tất.

Khi cài đặt hoàn tất, bạn sẽ thấy thông báo "Composer was successfully installed!" Điều này có nghĩa rằng Composer đã được cài đặt thành công trên hệ thống của bạn.

Bước 3: Kiểm tra cài đặt Composer

Mở Command Prompt hoặc PowerShell.

Để kiểm tra xem Composer đã được cài đặt thành công, chạy lệnh sau:

composer --version

Nếu bạn thấy phiên bản của Composer được hiển thị, điều đó có nghĩa là Composer đã được cài đặt thành công trên Windows của bạn.

Bây giờ bạn đã cài đặt Composer thành công và có thể sử dụng nó để quản lý các gói và thư viện PHP trong dự án của bạn trên Windows.

# Bài 2. Cài đặt laravel

**Bước 1:** Tạo một dự án Laravel mới

Mở Command Prompt hoặc PowerShell và di chuyển đến thư mục mà bạn muốn tạo dự án Laravel trong đó.

Sử dụng Composer để tạo một dự án Laravel mới. Chạy lệnh sau:

composer create-project --prefer-dist laravel/laravel project-name

Thay project-name bằng tên của dự án bạn muốn tạo.

Chờ đợi cho quá trình tạo dự án Laravel hoàn tất. Composer sẽ tải xuống tất cả các phụ thuộc và cấu hình cơ bản cho dự án.

**Bước 2:** Chạy dự án Laravel

Di chuyển đến thư mục dự án Laravel mới đã tạo:

cd project-name

Khởi động máy chủ phát triển bằng Laravel Artisan. Chạy lệnh sau:

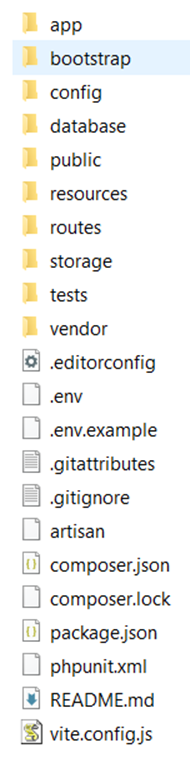
php artisan serve

Máy chủ sẽ khởi động và bạn sẽ nhận được URL để truy cập ứng dụng Laravel của bạn trên trình duyệt web.

Truy cập URL trên trình duyệt của bạn để xem dự án Laravel của bạn hoạt động.

# Bài 3. Các thành phần trong laravel

**A. Cấu trúc thư mục:**



Laravel có một cấu trúc thư mục và tập tin tiêu chuẩn giúp bạn dễ dàng tổ chức và quản lý dự án của mình. Dưới đây là một cái nhìn tổng quan về cấu trúc thư mục và tập tin trong Laravel:

**app**: Thư mục này chứa mã nguồn của ứng dụng Laravel. Nó bao gồm các thành phần chính như Controllers, Models, và Providers.

* **Http**: Thư mục này chứa các controllers, middleware, và requests.
* **Providers**: Thư mục này chứa các ServiceProvider của ứng dụng.

**bootstrap**: Chứa các tập tin cần thiết để khởi động ứng dụng Laravel.

**config**: Chứa các tập tin cấu hình cho ứng dụng, bao gồm cấu hình database, auth, và các tùy chọn khác.

**database**: Chứa các migrations và seeders để quản lý cơ sở dữ liệu.

* **migrations**: Thư mục này chứa các tập tin migration để tạo và cập nhật cơ sở dữ liệu.
* **seeders**: Thư mục này chứa các tập tin seeder để điền dữ liệu mẫu vào cơ sở dữ liệu.

**public**: Đây là thư mục root của ứng dụng và chứa các tài nguyên tĩnh như CSS, JavaScript, hình ảnh. Tệp index.php trong thư mục này là điểm khởi đầu của ứng dụng.

**resources**: Chứa các tài nguyên không phải mã nguồn như blade templates, các tệp CSS và JavaScript chưa được biên dịch.

* **assets**: Thư mục này chứa các tài nguyên gốc như tệp SCSS, LESS, và JavaScript không biên dịch.
* **views**: Thư mục này chứa các tập tin Blade templates.

**routes**: Chứa các tập tin route và tệp api.php, web.php để định tuyến các yêu cầu HTTP đến controllers hoặc các hàm xử lý khác.

**storage**: Chứa các tập tin tạm thời, như session, cache, và tệp tin đã tải lên. Thư mục này cũng chứa thư mục app, framework, và logs cho các mục đích lưu trữ khác.

**tests**: Chứa các test cho ứng dụng sử dụng PHPUnit.

**vendor**: Chứa các gói Composer đã tải xuống và các tệp mã nguồn của chúng.

**.env**: Tệp này chứa cấu hình môi trường cho ứng dụng, bao gồm cài đặt cơ sở dữ liệu, biến môi trường và các tùy chọn khác.

**.env.example**: Mẫu cấu hình môi trường. Bạn có thể sao chép và đổi tên thành .env để cấu hình ứng dụng của mình.

**.gitignore:** Tệp này chứa danh sách các tệp và thư mục mà Git sẽ bỏ qua khi bạn thực hiện các thao tác Git.

**composer.json:** Tệp này chứa các phụ thuộc và cấu hình cho Composer.

**README.md**: Tệp README cho dự án của bạn, chứa hướng dẫn và thông tin khác về dự án.

Cấu trúc thư mục và tập tin trong Laravel được thiết kế để giúp bạn dễ dàng quản lý và phát triển ứng dụng của mình.

**B. Các thành phần trong laravel**

Laravel là một framework phát triển ứng dụng web mạnh mẽ và được thiết kế để giúp bạn xây dựng các ứng dụng web hiện đại nhanh chóng và dễ dàng. Laravel bao gồm nhiều thành phần và tính năng mạnh mẽ, sau đây là một số thành phần quan trọng trong Laravel:

**Routing (Định tuyến)**: Laravel cung cấp một hệ thống định tuyến mạnh mẽ, cho phép bạn định tuyến các URL đến các điểm cuối ứng dụng của bạn và xác định các hàm xử lý tương ứng.

**Middleware:** Middleware là một lớp trung gian cho phép bạn thực hiện các xử lý trước và sau khi một yêu cầu được xử lý. Middleware có thể được sử dụng để xác thực, xử lý CORS, ghi nhật ký và nhiều tác vụ khác.

**Controllers (Bộ điều khiển):** Controllers trong Laravel cho phép bạn tổ chức mã xử lý cho các yêu cầu HTTP và quản lý logic ứng dụng.

Views (Giao diện): Laravel hỗ trợ Blade, một ngôn ngữ mẫu mạnh mẽ để tạo và quản lý giao diện người dùng. Blade cho phép bạn tạo các mẫu dễ đọc và dễ quản lý.

**Eloquent ORM:** Laravel đi kèm với một ORM (Object-Relational Mapping) mạnh mẽ gọi là Eloquent. Nó cho phép bạn làm việc với cơ sở dữ liệu bằng cách sử dụng các đối tượng và mô hình thay vì truy vấn SQL trực tiếp.

**Database Migrations:** Migrations là một cách để quản lý cơ sở dữ liệu và thực hiện các thay đổi cơ sở dữ liệu một cách dễ dàng và theo trình tự. Điều này giúp bạn duy trì cơ sở dữ liệu theo cách thống nhất giữa các môi trường.

**Eloquent Relationships:** Eloquent cho phép bạn quản lý quan hệ giữa các bảng cơ sở dữ liệu dễ dàng, chẳng hạn như quan hệ một-nhiều, nhiều-nhiều, và một-đến-một.

**Authentication và Authorization:** Laravel cung cấp các tính năng tích hợp để xác thực và phân quyền người dùng, giúp bảo vệ ứng dụng của bạn khỏi truy cập trái phép.

**Artisan Console:** Artisan là một công cụ dòng lệnh mạnh mẽ cho phép bạn thực hiện nhiều tác vụ quản lý ứng dụng như tạo controllers, models, migrations và nhiều tác vụ khác.

**Form Request và Validation:** Laravel cung cấp một cách tiện lợi để xác thực và xử lý dữ liệu gửi từ các biểu mẫu web.

**Task Scheduling:** Laravel cho phép bạn lên lịch thực thi các tác vụ theo định kỳ bằng cách sử dụng lệnh Artisan, giúp bạn tự động hóa các tác vụ như gửi email định kỳ hoặc xử lý dữ liệu.

**Testing (Kiểm thử):** Laravel hỗ trợ việc kiểm thử ứng dụng với PHPUnit và cung cấp các tiện ích cho việc kiểm tra đơn vị và kiểm thử tích hợp.

**File Storage và Uploads:** Laravel cung cấp hỗ trợ cho việc lưu trữ và quản lý tệp tin dễ dàng, bao gồm cả tải lên và truy cập vào tệp tin.

**API Support:** Laravel làm cho việc phát triển API dễ dàng thông qua tích hợp hỗ trợ cho API và đa dạng hóa đầu ra.

**Real-Time Broadcasting:** Laravel hỗ trợ tích hợp cho việc gửi thông báo thời gian thực sử dụng WebSockets thông qua Laravel Echo và Redis.

**Localization và Internationalization:** Laravel hỗ trợ đa ngôn ngữ và đa vùng giúp bạn xây dựng ứng dụng đa quốc gia.

# Bài 4. Làm việc với Database

## **1. Cấu hình kết nối CSDL**

Bước 1:<http://localhost/phpmyadmin/> và tạo tên CSDL: react\_laravel

Bước 2: Mở tập tin: .env

**Thiết lập tên cơ sở dữ liệu**

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=cdtt\_laravel

DB\_USERNAME=root

DB\_PASSWORD=

**Bước 3. Vào config/database.php**

'mysql' => [

//…

'prefix' => 'db\_', //Sửa tiền tố bảng

// …

],

## **2. Tạo Migration**

Để tạo một migration mới, bạn có thể sử dụng lệnh make:migration của Artisan. Để tạo một migration để tạo bảng có thể chạy lệnh sau:

php artisan make:migration create\_config\_table –create=db\_config

php artisan make:migration create\_brand\_table --create=db\_brand

php artisan make:migration create\_category\_table --create=db\_category

php artisan make:migration create\_contact\_table --create=db\_contact

php artisan make:migration create\_menu\_table --create=db\_menu

php artisan make:migration create\_order\_table --create=db\_order

php artisan make:migration create\_orderdetail\_table --create=db\_orderdetail

php artisan make:migration create\_product\_table --create=db\_product

php artisan make:migration create\_post\_table --create=db\_post

php artisan make:migration create\_banner\_table --create=db\_banner

php artisan make:migration create\_topic\_table --create=db\_topic

php artisan make:migration create\_user\_table --create=db\_user

php artisan make:migration create\_productstore\_table --create=db\_productstore

php artisan make:migration create\_productsale\_table --create=db\_productsale

Sau khi chạy lệnh này, Laravel sẽ tạo ra một tập tin migration mới trong thư mục database/migrations. Bạn có thể mở tập tin này để chỉnh sửa nó.

## **3. Định nghĩa Schema**

Trong tập tin migration, bạn có thể sử dụng các phương thức của lớp Schema để định nghĩa cấu trúc của bảng cơ sở dữ liệu.

### **3.1. Bảng db\_banner**

public function up(): void

{

Schema::create('db\_banner', function (Blueprint $table) {

$table->id(); //id

$table->string('name', 255);

$table->string('image', 255);

$table->string('link', 1000);

$table->string('position', 255);

$table->string('description', 1000);

$table->unsignedInteger('created\_by')->default(1);

$table->unsignedInteger('updated\_by')->nullable();

$table->timestamps();

$table->unsignedTinyInteger('status')->default(2);

});

}

### **3.2. Bảng db\_brand**

public function up(): void

{

Schema::create('db\_brand', function (Blueprint $table) {

$table->id(); //id

$table->string('name', 1000);

$table->string('slug', 1000);

$table->string('image', 1000)->nullable();

$table->unsignedInteger('sort\_order')->default(0);

$table->string('description',1000);

$table->timestamps(); //created\_at, updated\_at

$table->unsignedInteger('created\_by')->default(1);

$table->unsignedInteger('updated\_by')->nullable();

$table->unsignedTinyInteger('status')->default(2);

});

}

### **3.3. Bảng db\_category**

public function up(): void

{

Schema::create('db\_category', function (Blueprint $table) {

$table->id(); //id

$table->string('name', 1000);

$table->string('slug', 1000);

$table->string('image', 1000)->nullable();

$table->unsignedInteger('parent\_id')->default(0);

$table->unsignedInteger('sort\_order')->default(0);

$table->string('description',1000);

$table->timestamps(); //created\_at, updated\_at

$table->unsignedInteger('created\_by')->default(1);

$table->unsignedInteger('updated\_by')->nullable();

$table->unsignedTinyInteger('status')->default(2);

});

}

### **3.4: Bảng db\_config**

public function up(): void

{

Schema::create('db\_config', function (Blueprint $table) {

$table->id(); //id

$table->string('author');

$table->string('email');

$table->string('phone');

$table->string('zalo');

$table->string('facebook');

$table->string('address');

$table->string('youtube');

$table->string('metadesc');

$table->string('metakey');

$table->timestamps(); //created\_at, updated\_at

$table->unsignedInteger('created\_by')->default(1);

$table->unsignedInteger('updated\_by')->nullable();

$table->unsignedTinyInteger('status')->default(2);

});

}

### **3.5. Bảng db\_contact**

public function up(): void

{

Schema::create('db\_contact', function (Blueprint $table) {

$table->id(); //id

$table->unsignedInteger('user\_id')->nullable();

$table->string('name');

$table->string('email');

$table->string('phone');

$table->string('title');

$table->mediumText('content');

$table->unsignedInteger('replay\_id')->nullable();

$table->timestamps(); //created\_at, updated\_at

$table->unsignedInteger('created\_by')->default(1);

$table->unsignedInteger('updated\_by')->nullable();

$table->unsignedTinyInteger('status')->default(2);

});

}

### **3.6. Bảng db\_menu**

public function up(): void

{

Schema::create('db\_menu', function (Blueprint $table) {

$table->id(); //id

$table->string('name', 1000);

$table->string('link', 1000);

$table->unsignedInteger('sort\_order')->default(0);

$table->unsignedInteger('parent\_id')->default(0);

$table->string('type', 100)->nullable();

$table->unsignedInteger('table\_id')->default(0);

$table->string('description',1000);

$table->timestamps(); //created\_at, updated\_at

$table->unsignedInteger('created\_by')->default(1);

$table->unsignedInteger('updated\_by')->nullable();

$table->unsignedTinyInteger('status')->default(2);

});

}

### **3.7. Bảng db\_order**

public function up(): void

{

Schema::create('db\_order', function (Blueprint $table) {

$table->id(); //id

$table->unsignedInteger('user\_id');

$table->string('delivery\_name', 255);

$table->string('delivery\_gender', 255);

$table->string('delivery\_email', 255);

$table->string('delivery\_phone', 255);

$table->string('delivery\_address', 1000);

$table->string('note', 1000);

$table->timestamps(); //created\_at, updated\_at

$table->unsignedInteger('created\_by')->default(1);

$table->unsignedInteger('updated\_by')->nullable();

$table->unsignedTinyInteger('status')->default(2);

});

}

### **3.8. Bảng db\_orderdetail**

public function up(): void

{

Schema::create('db\_orderdetail', function (Blueprint $table) {

$table->id(); //id

$table->unsignedInteger('order\_id');

$table->unsignedInteger('product\_id');

$table->double('price');

$table->unsignedInteger('qty');

$table->double('discount');

$table->double('amount');

});

}

### **3.9. Bảng db\_post**

public function up(): void

{

Schema::create('db\_post', function (Blueprint $table) {

$table->id(); //id

$table->unsignedInteger('topic\_id')->nullable();

$table->string('title', 1000);

$table->string('slug', 1000);

$table->text('detail');

$table->string('description',255);

$table->string('image',255);

$table->string('type',255);

$table->timestamps(); //created\_at, updated\_at

$table->unsignedInteger('created\_by')->default(1);

$table->unsignedInteger('updated\_by')->nullable();

$table->unsignedTinyInteger('status')->default(2);

});

}

### **3.10. Bảng db\_product**

public function up(): void

{

Schema::create('db\_product', function (Blueprint $table) {

$table->id(); //id

$table->unsignedInteger('category\_id');

$table->unsignedInteger('brand\_id');

$table->string('name', 1000);

$table->string('slug', 1000);

$table->text('detail');

$table->string('description',255);

$table->string('image',1000);

$table->double('price');

$table->timestamps(); //created\_at, updated\_at

$table->unsignedInteger('created\_by')->default(1);

$table->unsignedInteger('updated\_by')->nullable();

$table->unsignedTinyInteger('status')->default(2);

});

}

### **3.11. Bảng db\_productsale**

public function up(): void

{

Schema::create('db\_productsale', function (Blueprint $table) {

$table->id(); //id

$table->unsignedInteger('product\_id');

$table->double('pricesale');

$table->unsignedInteger('qty');

$table->datetime('date\_begin');

$table->datetime('date\_end');

$table->timestamps(); //created\_at, updated\_at

$table->unsignedInteger('created\_by')->default(1);

$table->unsignedInteger('updated\_by')->nullable();

});

}

### **3.12. Bảng db\_productstore**

public function up(): void

{

Schema::create('db\_productstore', function (Blueprint $table) {

$table->id(); //id

$table->unsignedInteger('product\_id');

$table->double('price');

$table->unsignedInteger('qty');

$table->timestamps(); //created\_at, updated\_at

$table->unsignedInteger('created\_by')->default(1);

$table->unsignedInteger('updated\_by')->nullable();

});

}

### **3.13. Bảng db\_topic**

public function up(): void

{

Schema::create('db\_topic', function (Blueprint $table) {

$table->id(); //id

$table->string('name', 1000);

$table->string('slug', 1000);

$table->unsignedInteger('sort\_order')->default(0);

$table->string('description',1000);

$table->timestamps(); //created\_at, updated\_at

$table->unsignedInteger('created\_by')->default(1);

$table->unsignedInteger('updated\_by')->nullable();

$table->unsignedTinyInteger('status')->default(2);

});

}

### **3.14. Bảng db\_user**

public function up(): void

{

Schema::create('db\_contact', function (Blueprint $table) {

$table->id(); //id

$table->string('name');

$table->string('username');

$table->string('password');

$table->string('gender');

$table->string('phone');

$table->string('email');

$table->string('roles');

$table->timestamps(); //created\_at, updated\_at

$table->unsignedInteger('created\_by')->default(1);

$table->unsignedInteger('updated\_by')->nullable();

$table->unsignedTinyInteger('status')->default(2);

});

}

## **4. Chạy Migration**

Để áp dụng migration và tạo bảng cơ sở dữ liệu, bạn có thể chạy lệnh migrate:

php artisan migrate

# Bài 5. Tạo model

## **1. Tạo các tập tin Model(app/Models)**

(https://laravel.com/docs/10.x/eloquent)

- Mở cmd.exe

- Chạy từng lệnh sau:

php artisan make:model Banner

php artisan make:model Brand

php artisan make:model Category

php artisan make:model Config

php artisan make:model Contact

php artisan make:model Menu

php artisan make:model Order

php artisan make:model Orderdetail

php artisan make:model Post

php artisan make:model Product

php artisan make:model ProductSale

php artisan make:model ProductStore

php artisan make:model Topic

php artisan make:model User

## **2. Khai báo từng model**

### **2.1. Banner**

class Banner extends Model

{

use HasFactory;

protected $table='banner';

}

### **2.2. Brand**

class Brand extends Model

{

use HasFactory;

protected $table='brand';

}

### **2.3. Category**

class Category extends Model

{

use HasFactory;

protected $table='category';

}

### **2.4. Config**

class Config extends Model

{

use HasFactory;

protected $table='config';

}

### **2.5. Contact**

class Contact extends Model

{

use HasFactory;

protected $table='contact';

}

### **2.6. Menu**

class Menu extends Model

{

use HasFactory;

protected $table='menu';

}

### **2.7. Order**

class Order extends Model

{

use HasFactory;

protected $table='order';

}

### **2.8. Orderdetail**

class Orderdetail extends Model

{

use HasFactory;

protected $table='orderdetail';

public $timestamps = false;

}

### **2.9. Post**

class Post extends Model

{

use HasFactory;

protected $table='post';

}

### **2.10. Product**

class Product extends Model

{

use HasFactory;

protected $table='product';

}

### **2.11. ProductSale**

class ProductSale extends Model

{

use HasFactory;

protected $table='productsale';

}

### **2.12. ProductStore**

class ProductStore extends Model

{

use HasFactory;

protected $table='productstore';

}

### **2.13. Topic**

class Topic extends Model

{

use HasFactory;

protected $table='topic';

}

### **2.14. User**

class User extends Authenticatable

{

use HasApiTokens, HasFactory, Notifiable;

protected $table='user';

protected $fillable = [

'name',

'email',

'password',

];

protected $hidden = [

'password',

'remember\_token',

];

protected $casts = [

'email\_verified\_at' => 'datetime',

'password' => 'hashed',

];

}

# Bài 6. Tạo controller

Bước 1. Tạo controller (app\http\Controllers)

- Mở cmd.exe

- Chạy từng lệnh sau

php artisan make:controller BrandController

php artisan make:controller CategoryController

php artisan make:controller ContactController

php artisan make:controller MenuController

php artisan make:controller OrderController

php artisan make:controller OrderdetailController

php artisan make:controller ProductController

php artisan make:controller PostController

php artisan make:controller BannerController

php artisan make:controller TopicController

php artisan make:controller UserController

# Bài 7. Khai báo các route

**1. Vị trí của route**

routes/web.php

**2. Các loại phương thức**

GET → Route::get()

POST → Route::post()

# Bài 8. Đăng nhập hệ thống, quên mật khẩu

## **1. Đăng nhập**

**Bước 1. Tổng quan**

* Phương thức: get
* Url: admin/login
* Controller: backend/AuthControlller
* Hàm (function): login

**Bước 2. Khai báo route (routes/web.php)**

use App\Http\Controllers\backend\AuthController;

Route::get('admin/login', [AuthController::class, 'login'])->route(‘admin.login’);

**Bước 3. Viết mã lệnh**

public function login()

{

$categorys = Category::all();

return response()->json(

[

'success' => true,

'message' => 'Tải dữ liệu thành công',

'categorys' => $categorys

],

200

);

}

## **2. Quên mật khẩu**

ádsad

# Bài 9. Cấu hình website(config)

**Bước 1. Tổng quan**

* Phương thức: get
* Url: admin/config
* Controller: backend/ConfigControlller
* Hàm (function): index

**Bước 2. Khai báo route (routes/web.php)**

use App\Http\Controllers\backend\CategoryController;

Route::get('config/index', [ConfigController::class, 'index']);

**Bước 3. Viết mã lệnh truy vấn dữ liệu và truyền dữ liệu sang view**

public function index()

{

$config = Category::first();

return view(‘backend.config.index’, compact(‘config’));

}

[Học viên: tự tạo view tương ứng]

**Bước 4. Viết mã lệnh xử lý cập nhật**

public function update(Request $request, string $id)

{

$config = Category::find($id);

$config->site\_name = $request->site\_name;

$config->save();

}

# Bài 10. Quản lý danh mục sản phẩm

**1. Hiển thị danh sách danh mục**

**Bước 1. Tổng quan**

* Phương thức: get
* Url: admin/category/index
* Controller: backend/CategoryControlller
* Hàm (function): index

**Bước 2. Khai báo route (routes/web.php)**

use App\Http\Controllers\backend\CategoryController;

Route::get('category/index', [CategoryController::class, 'index']);

**Bước 3. Viết mã lệnh**

public function index()

{

$categorys = Category::where(‘status’, ‘!=’,0)

->orderBy(‘created\_at’,‘desc’)

->get();

return view(‘backend.category.index’, compact(‘categorys’));

}

**2. Hiển thị danh sách thùng rác loại**

**Bước 1. Tổng quan**

* Phương thức: get
* Url: admin/category/trash
* Controller: backend/CategoryControlller
* Hàm (function): trash

**Bước 2. Khai báo route (routes/web.php)**

use App\Http\Controllers\backend\CategoryController;

Route::get('category/trash', [CategoryController::class, 'trash']);

**Bước 3. Viết mã lệnh**

public function index()

{

$categorys = Category::where(‘status’, ‘=’,0)

->orderBy(‘created\_at’,‘desc’)

->get();

return view(‘backend.category.trash’, compact(‘categorys’));

}

**3. Hiển thị chi tiết**

**Bước 1. Tổng quan**

* Phương thức: get
* Url: admin/category/show/1
* Controller: backend/CategoryControlller
* Hàm (function): show

**Bước 2. Khai báo route (routes/web.php)**

use App\Http\Controllers\Api\CategoryController;

Route::get('category/show/{id}', [CategoryController::class, 'show']);

**Bước 3. Viết mã lệnh**

public function show($id)

{

$category = Category::find($id);

if($category==null)

{

return redictect()->route(‘’);

}

return view(‘backend.category.show’, compact(‘category’));

}

**4. Thêm**

**Bước 1. Tổng quan**

* Phương thức: post
* Url: api/category/store
* Controller: backend/CategoryControlller
* Hàm (function): store

**Bước 2. Khai báo route api (routes/web.php)**

use App\Http\Controllers\backend\CategoryController;

Route::post('category/store', [CategoryController::class, 'store']);

**Bước 3. Viết mã lệnh**

public function store(Request $request)

{

$category = new Category();

$category->name = $request->name; //form

$category->slug = Str::of($request->name)->slug('-');

//upload image

$files = $request->image;

if ($files != null)

{

$extension = $files->getClientOriginalExtension();

if (in\_array($extension, ['jpg', 'png', 'gif', 'webp', 'jpeg']))

{

$filename = $category->slug . '.' . $extension;

$category->image = $filename;

$files->move(public\_path('images/category'), $filename);

}

}

//

$category->sort\_order = $request->sort\_order; //form

$category->metakey = $request->metakey; //form

$category->metadesc = $request->metadesc; //form

$category->created\_at = date('Y-m-d H:i:s');

$category->created\_by = 1;

$category->status = $request->status; //form

$category->save();

return response()->json(

[

'success' => true,

'message' => 'Thêm danh mục thành công',

'category' => $category

],

201

);

}

**5. Cập nhật**

**Bước 1. Khai báo**

* Phương thức: post
* Url: api/category/update
* Controller: backend/CategoryControlller
* Hàm (function): update

**Bước 2. Khai báo route api (routes/web.php)**

use App\Http\Controllers\Api\CategoryController;

Route::post('category/update/{id}', [CategoryController::class, 'update']);

**Bước 3. Viết mã lệnh**

public function update(Request $request, $id)

{

$category = Category::find($id);

$category->name = $request->name; //form

$category->slug = Str::of($request->name)->slug('-');

//upload image

$files = $request->image;

if ($files != null)

{

$extension = $files->getClientOriginalExtension();

if (in\_array($extension, ['jpg', 'png', 'gif', 'webp', 'jpeg']))

{

$filename = $category->slug . '.' . $extension;

$category->image = $filename;

$files->move(public\_path('images/brand'), $filename);

}

}

//

$category->sort\_order = $request->sort\_order; //form

$category->metakey = $request->metakey; //form

$category->metadesc = $request->metadesc; //form

$category->updated\_at = date('Y-m-d H:i:s');

$category->updated\_by = 1;

$category->status = $request->status; //form

$category->save(); //Luuu vao CSDL

return response()->json(

[

'success' => true,

'message' => 'Thành công',

'category' => $category

],

200

);

}

**6. Thay đổi trạng thái**

**7. Xóa vào thùng rác**

**8. Khôi phục rác**

**9. Xóa khỏi CSDL**

**Bước 1. Khai báo**

* Phương thức: delete
* Url: api/category/destroy/4
* Controller: backend/CategoryControlller
* Hàm (function): destroy

**Bước 2. Khai báo route api (routes/web.php)**

use App\Http\Controllers\Api\CategoryController;

Route::delete('category/destroy/{id}', [CategoryController::class, 'destroy']);

**Bước 3. Viết mã lệnh**

public function destroy($id)

{

$category = Brand::findOrFail($id);

$category->delete();

return response()->json(

[

'success' => true,

'message' => 'Xóa thành công',

'category' => null

],

200

);

}

# Bài 11. Quản lý thương hiệu

## **1. Hiển thị danh sách danh mục**

**Bước 1. Tổng quan**

* Phương thức: get
* Url: admin/brand/index
* Controller: backend/BrandControlller
* Hàm (function): index

**Bước 2. Khai báo route (routes/web.php)**

use App\Http\Controllers\backend\BrandController;

Route::get('brand/index', [BrandController::class, 'index']);

**Bước 3. Viết mã lệnh**

public function index()

{

$brands = Brand::all();

return response()->json(

[

'success' => true,

'message' => 'Tải dữ liệu thành công',

'brands' => $brands

],

200

);

}

## **2. Hiển thị danh sách thùng rác loại**

sdf

## **3. Thêm**

**Bước 1. Tổng quan**

* Phương thức: post
* Url: api/brand/store
* Controller: backend/BrandControlller
* Hàm (function): store

**Bước 2. Khai báo route api (routes/web.php)**

use App\Http\Controllers\backend\BrandController;

Route::post('brand/store', [BrandController::class, 'store']);

**Bước 3. Viết chức năng thêm BrandController/store**

public function store(Request $request)

{

$brand = new Brand();

$brand->name = $request->name; //form

$brand->slug = Str::of($request->name)->slug('-');

//upload image

$files = $request->image;

if ($files != null)

{

$extension = $files->getClientOriginalExtension();

if (in\_array($extension, ['jpg', 'png', 'gif', 'webp', 'jpeg']))

{

$filename = $brand->slug . '.' . $extension;

$brand->image = $filename;

$files->move(public\_path('images/brand'), $filename);

}

}

//

$brand->sort\_order = $request->sort\_order; //form

$brand->metakey = $request->metakey; //form

$brand->metadesc = $request->metadesc; //form

$brand->created\_at = date('Y-m-d H:i:s');

$brand->created\_by = 1;

$brand->status = $request->status; //form

$brand->save();

return response()->json(

[

'success' => true,

'message' => 'Thêm thương hiệu thành công',

'brand' => $brand

],

201

);

}

## **4. Chi tiết**

**Bước 1. Tổng quan**

* Phương thức: get
* Url: admin/brand/show/1
* Controller: backend/BrandControlller
* Hàm (function): show

**Bước 2. Khai báo route api (routes/web.php)**

use App\Http\Controllers\backend\BrandController;

Route::get('brand/show/{id}', [BrandController::class, 'show']);

**Bước 3. Viết mã lệnh**

public function show($id)

{

$brand = Brand::find($id);

if($brand==null)

{

return response()->json(

[

'success' => false,

'message' => 'Không tìm thấy thông tin',

'brand' => NULL

],

200

);

}

return response()->json(

[

'success' => true,

'message' => 'Tải dữ liệu thành công',

'brand' => $brand

],

200

);

}

## **5. Cập nhật**

**Bước 1. Khai báo**

* Phương thức: post
* Url: api/brand/update
* Controller: Api/BrandControlller
* Hàm (function): update

**Bước 2. Khai báo route api (routes/web.php)**

use App\Http\Controllers\Api\BrandController;

Route::post('brand/update/{id}', [BrandController::class, 'update']);

**Bước 3. Viết mã lệnh**

public function update(Request $request, $id)

{

$brand = Brand::find($id);

$brand->name = $request->name; //form

$brand->slug = Str::of($request->name)->slug('-');

//upload image

$files = $request->image;

if ($files != null)

{

$extension = $files->getClientOriginalExtension();

if (in\_array($extension, ['jpg', 'png', 'gif', 'webp', 'jpeg']))

{

$filename = $brand->slug . '.' . $extension;

$brand->image = $filename;

$files->move(public\_path('images/brand'), $filename);

}

}

//

$brand->sort\_order = $request->sort\_order; //form

$brand->metakey = $request->metakey; //form

$brand->metadesc = $request->metadesc; //form

$brand->updated\_at = date('Y-m-d H:i:s');

$brand->updated\_by = 1;

$brand->status = $request->status; //form

$brand->save(); //Luuu vao CSDL

return response()->json(

[

'success' => true,

'message' => 'Thành công',

'brand' => $brand

],

200

);

}

## **6. Thay đổi trạng thái**

áds

## **7. Xóa vào thùng rác**

áds

## **8. Khôi phục rác**

adssd

## **9. Xóa khỏi CSDL**

**Bước 1. Khai báo**

* Phương thức: delete
* Url: api/brand/destroy/4
* Controller: Api/BrandControlller
* Hàm (function): destroy

**Bước 2. Khai báo route api (routes/web.php)**

use App\Http\Controllers\Api\BrandController;

Route::delete('brand/destroy/{id}', [BrandController::class, 'destroy']);

**Bước 3. Viết mã lệnh**

public function destroy($id)

{

$brand = Brand::findOrFail($id);

$brand->delete();

return response()->json(

[

'success' => true,

'message' => 'Xóa thành công',

'brand' => null

],

200

);

}

## 

# Bài 12. Quản lý sản phẩm

## **1. Hiển thị danh sách danh mục**

**Bước 1. Tổng quan**

* Phương thức: get
* Url: admin/product/index
* Controller: backend/ProductControlller
* Hàm (function): index

**Bước 2. Khai báo route api (routes/web.php)**

use App\Http\Controllers\backend\ProductController;

Route::get('product/index', [ProductController::class, 'index']);

**Bước 3. Viết mã lệnh**

public function index()

{

$products = Product::where( ‘status’, ‘!=’, 0)

->orderBy(‘created\_at’, ‘desc’)

->get();

return view(‘backend.product.index’,compact(‘products’));

}

## **2. Hiển thị danh sách thùng rác loại**

sdf

## **3. Thêm**

**Bước 1. Tổng quan**

* Phương thức: post
* Url: api/product/store
* Controller: Api/ProductControlller
* Hàm (function): store

**Bước 2. Khai báo route api (routes/web.php)**

use App\Http\Controllers\Api\ProductController;

Route::post('product/store', [ProductController::class, 'store']);

**Bước 3. Viết mã lệnh**

public function store(Request $request)

{

$product = new Category();

$product->name = $request->name; //form

$product->slug = Str::of($request->name)->slug('-');

//upload image

$files = $request->image;

if ($files != null)

{

$extension = $files->getClientOriginalExtension();

if (in\_array($extension, ['jpg', 'png', 'gif', 'webp', 'jpeg']))

{

$filename = $category->slug . '.' . $extension;

$category->image = $filename;

$files->move(public\_path('images/category'), $filename);

}

}

//

$product->sort\_order = $request->sort\_order; //form

$product->metakey = $request->metakey; //form

$product->metadesc = $request->metadesc; //form

$product->created\_at = date('Y-m-d H:i:s');

$product->created\_by = 1;

$product->status = $request->status; //form

$product->save();

return response()->json(

[

'success' => true,

'message' => 'Thêm sản phẩm thành công',

'product' => $product

],

201

);

}

## **4. Chi tiết**

**Bước 1. Tổng quan**

* Phương thức: get
* Url: api/category/show/1
* Controller: Api/ProductControlller
* Hàm (function): show

**Bước 2. Khai báo route api (routes/web.php)**

use App\Http\Controllers\Api\ProductController;

Route::get('product/show/{id}', [ProductController::class, 'show']);

**Bước 3. Viết mã lệnh**

public function show($id)

{

$product = Category::find($id);

if($product==null)

{

return response()->json(

[

'success' => false,

'message' => 'Không tìm thấy kết quả',

'category' => NULL

],

200

);

}

return response()->json(

[

'success' => true,

'message' => 'Tải dữ liệu thành công',

'product' => $product

],

200

);

}

## **5. Cập nhật**

**Bước 1. Khai báo**

* Phương thức: post
* Url: api/product/update
* Controller: Api/ProductControlller
* Hàm (function): update

**Bước 2. Khai báo route api (routes/web.php)**

use App\Http\Controllers\Api\CategoryController;

Route::post('brand/update/{id}', [CategoryController::class, 'update']);

**Bước 3. Viết mã lệnh**

public function update(Request $request, $id)

{

$category = Category::find($id);

$category->name = $request->name; //form

$category->slug = Str::of($request->name)->slug('-');

//upload image

$files = $request->image;

if ($files != null)

{

$extension = $files->getClientOriginalExtension();

if (in\_array($extension, ['jpg', 'png', 'gif', 'webp', 'jpeg']))

{

$filename = $category->slug . '.' . $extension;

$category->image = $filename;

$files->move(public\_path('images/brand'), $filename);

}

}

//

$category->sort\_order = $request->sort\_order; //form

$category->metakey = $request->metakey; //form

$category->metadesc = $request->metadesc; //form

$category->updated\_at = date('Y-m-d H:i:s');

$category->updated\_by = 1;

$category->status = $request->status; //form

$category->save(); //Luuu vao CSDL

return response()->json(

[

'success' => true,

'message' => 'Thành công',

'category' => $category

],

200

);

}

## **6. Thay đổi trạng thái**

áds

## **7. Xóa vào thùng rác**

áds

## **8. Khôi phục rác**

adssd

## **9. Xóa khỏi CSDL**

**Bước 1. Khai báo**

* Phương thức: delete
* Url: api/category/destroy/4
* Controller: backend/CategoryControlller
* Hàm (function): destroy

**Bước 2. Khai báo route api (routes/web.php)**

use App\Http\Controllers\Api\CategoryController;

Route::delete('category/destroy/{id}', [CategoryController::class, 'destroy']);

**Bước 3. Viết mã lệnh**

public function destroy($id)

{

$category = Brand::findOrFail($id);

$category->delete();

return response()->json(

[

'success' => true,

'message' => 'Xóa thành công',

'category' => null

],

200

);

}

# Bài 13. Quản lý chủ đề bài viết

## **1. Hiển thị danh sách danh mục**

**Bước 1. Tổng quan**

* Phương thức: get
* Url: admin/topic/index
* Controller: backend/TopicControlller
* Hàm (function): index

**Bước 2. Khai báo route (routes/web.php)**

use App\Http\Controllers\backend\TopicController;

Route::get('topic/index', [TopicController::class, 'index']);

**Bước 3. Viết mã lệnh**

public function index()

{

$topics = Topic::where(‘status’, ‘!=’,0)

->orderBy(‘created\_at’,‘desc’)

->get();

return view(‘backend.topic.index’, compact(‘topics’));

}

## **2. Hiển thị danh sách thùng rác loại**

sdf

## **3. Thêm**

sds

## **4. Chi tiết**

adsa

## **5. Cập nhật**

sads

## **6. Thay đổi trạng thái**

áds

## **7. Xóa vào thùng rác**

áds

## **8. Khôi phục rác**

adssd

## **9. Xóa khỏi CSDL**

ád

# Bài 14. Quản lý bài viết

## **1. Hiển thị danh sách bài viết**

**Bước 1. Tổng quan**

* Phương thức: get
* Url: admin/post/index
* Controller: backend/PostControlller
* Hàm (function): index

**Bước 2. Khai báo route (routes/web.php)**

use App\Http\Controllers\backend\PostController;

Route::get('post/index', [PostController::class, 'index']);

**Bước 3. Viết mã lệnh**

public function index()

{

$posts = Post::where([[‘status’, ‘!=’,0],[‘type’, ‘=’, ‘post’]])

->orderBy(‘created\_at’,‘desc’)

->get();

return view(‘backend.post.index’, compact(‘posts’));

}

## **2. Hiển thị danh sách thùng rác loại**

sdf

## **3. Thêm**

sds

## **4. Chi tiết**

adsa

## **5. Cập nhật**

sads

## **6. Thay đổi trạng thái**

áds

## **7. Xóa vào thùng rác**

áds

## **8. Khôi phục rác**

adssd

## **9. Xóa khỏi CSDL**

ád

# Bài 15. Quản lý trang đơn

## **1. Hiển thị danh sách danh mục**

**Bước 1. Tổng quan**

* Phương thức: get
* Url: admin/page/index
* Controller: backend/PageControlller
* Hàm (function): index

**Bước 2. Khai báo route (routes/web.php)**

use App\Http\Controllers\backend\PageController;

Route::get('page/index', [PageController::class, 'index']);

**Bước 3. Viết mã lệnh**

public function index()

{

$pages = Post::where([[‘status’, ‘!=’,0],[‘type’, ‘=’,‘page’]])

->orderBy(‘created\_at’,‘desc’)

->get();

return view(‘backend.page.index’, compact(‘pages’));

}

## **2. Hiển thị danh sách thùng rác loại**

sdf

## **3. Thêm**

sds

## **4. Chi tiết**

adsa

## **5. Cập nhật**

sads

## **6. Thay đổi trạng thái**

áds

## **7. Xóa vào thùng rác**

áds

## **8. Khôi phục rác**

adssd

## **9. Xóa khỏi CSDL**

ád

# Bài 16. Quản lý menu(menu)

## **1. Hiển thị danh sách danh mục**

**Bước 1. Tổng quan**

* Phương thức: get
* Url: admin/menu/index
* Controller: backend/MenuControlller
* Hàm (function): index

**Bước 2. Khai báo route (routes/web.php)**

use App\Http\Controllers\backend\MenuController;

Route::get('menu/index', [MenuController::class, 'index']);

**Bước 3. Viết mã lệnh**

public function index()

{

$menus = Menu::where(‘status’, ‘!=’,0)

->orderBy(‘created\_at’,‘desc’)

->get();

$brands = Brand::where(‘status’, ‘!=’,0)

->orderBy(‘created\_at’,‘desc’)

->get();

$categorys = Category::where(‘status’, ‘!=’,0)

->orderBy(‘created\_at’,‘desc’)

->get();

$pages = Post::where([[‘status’, ‘!=’,0],[‘type’, ‘=’,‘page’]])

->orderBy(‘created\_at’,‘desc’)

->get();

$topics = Topic::where(‘status’, ‘!=’,0)

->orderBy(‘created\_at’,‘desc’)

->get();

return view(‘backend.menu.index’, compact(‘menus’, ‘brands’, ‘categorys’, ‘topics’));

}

## **2. Hiển thị danh sách thùng rác loại**

sdf

## **3. Thêm**

sds

## **4. Chi tiết**

adsa

## **5. Cập nhật**

sads

## **6. Thay đổi trạng thái**

áds

## **7. Xóa vào thùng rác**

áds

## **8. Khôi phục rác**

adssd

## **9. Xóa khỏi CSDL**

ád

# Bài 17. Quản lý banner(banner)

## **1. Hiển thị danh sách danh mục**

**Bước 1. Tổng quan**

* Phương thức: get
* Url: admin/banner/index
* Controller: backend/BannerControlller
* Hàm (function): index

**Bước 2. Khai báo route (routes/web.php)**

use App\Http\Controllers\backend\BannerController;

Route::get('banner/index', [BannerController::class, 'index']);

**Bước 3. Viết mã lệnh**

public function index()

{

$banners = Banner::where(‘status’, ‘!=’,0)

->orderBy(‘created\_at’,‘desc’)

->get();

return view(‘backend.banner.index’, compact(‘banners’));

}

## **2. Hiển thị danh sách thùng rác loại**

sdf

## **3. Thêm**

sds

## **4. Chi tiết**

adsa

## **5. Cập nhật**

sads

## **6. Thay đổi trạng thái**

áds

## **7. Xóa vào thùng rác**

áds

## **8. Khôi phục rác**

adssd

## **9. Xóa khỏi CSDL**

ád

# Bài 18. Quản lý đơn hàng(order & orderdetail)

## **1. Hiển thị danh sách đơn hàng**

**Bước 1. Tổng quan**

* Phương thức: get
* Url: admin/order/index
* Controller: backend/OrderControlller
* Hàm (function): index

**Bước 2. Khai báo route (routes/web.php)**

use App\Http\Controllers\backend\OrderController;

Route::get('order/index', [OrderController::class, 'index']);

**Bước 3. Viết mã lệnh**

public function index()

{

$orders = Order::where(‘status’, ‘!=’,0)

->orderBy(‘created\_at’,‘desc’)

->get();

return view(‘backend.order.index’, compact(‘orders’));

}

## **2. Hiển thị danh sách thùng rác loại**

sdf

## **3. Thêm**

sds

## **4. Chi tiết**

adsa

## **5. Cập nhật**

sads

## **6. Thay đổi trạng thái**

áds

## **7. Xóa vào thùng rác**

áds

## **8. Khôi phục rác**

adssd

## **9. Xóa khỏi CSDL**

ád

# Bài 19. Quản lý liên hệ (contact)

## **1. Hiển thị danh sách danh mục**

**Bước 1. Tổng quan**

* Phương thức: get
* Url: admin/contact/index
* Controller: backend/ContactControlller
* Hàm (function): index

**Bước 2. Khai báo route (routes/web.php)**

use App\Http\Controllers\backend\ContactController;

Route::get('contact/index', [ContactController::class, 'index']);

**Bước 3. Viết mã lệnh**

public function index()

{

$contacts = Contact::where(‘status’, ‘!=’,0)

->orderBy(‘created\_at’,‘desc’)

->get();

return view(‘backend.contact.index’, compact(‘contacts’));

}

## **2. Hiển thị danh sách thùng rác loại**

sdf

## **3. Thêm**

sds

## **4. Chi tiết**

adsa

## **5. Cập nhật**

sads

## **6. Thay đổi trạng thái**

áds

## **7. Xóa vào thùng rác**

áds

## **8. Khôi phục rác**

adssd

## **9. Xóa khỏi CSDL**

ád

# Bài 20. Quản lý khách hàng (user)

## **1. Hiển thị danh sách danh mục**

**Bước 1. Tổng quan**

* Phương thức: get
* Url: admin/customer/index
* Controller: backend/CustomerControlller
* Hàm (function): index

**Bước 2. Khai báo route (routes/web.php)**

use App\Http\Controllers\backend\CustomerController;

Route::get('customer/index', [CustomerController::class, 'index']);

**Bước 3. Viết mã lệnh**

public function index()

{

$customers = User::where([[‘status’, ‘!=’,0],[‘roles’, ‘=’,0]])

->orderBy(‘created\_at’,‘desc’)

->get();

return view(‘backend.customer.index’, compact(‘customers’));

}

## **2. Hiển thị danh sách thùng rác loại**

sdf

## **3. Thêm**

sds

## **4. Chi tiết**

adsa

## **5. Cập nhật**

sads

## **6. Thay đổi trạng thái**

áds

## **7. Xóa vào thùng rác**

áds

## **8. Khôi phục rác**

adssd

## **9. Xóa khỏi CSDL**

ád

# Bài 21. Quản lý thành viên(user)

## **1. Hiển thị danh sách thành viên**

**Bước 1. Tổng quan**

* Phương thức: get
* Url: admin/user/index
* Controller: backend/UserControlller
* Hàm (function): index

**Bước 2. Khai báo route (routes/web.php)**

use App\Http\Controllers\backend\UserController;

Route::get('user/index', [UserController::class, 'index']);

**Bước 3. Viết mã lệnh**

public function index()

{

$users = Category::where([[‘status’, ‘!=’,0],[‘roles’, ‘!=’,0]])

->orderBy(‘created\_at’,‘desc’)

->get();

return view(‘backend.user.index’, compact(‘users’));

}

## **2. Hiển thị danh sách thùng rác loại**

sdf

## **3. Thêm**

sds

## **4. Chi tiết**

adsa

## **5. Cập nhật**

sads

## **6. Thay đổi trạng thái**

áds

## **7. Xóa vào thùng rác**

áds

## **8. Khôi phục rác**

adssd

## **9. Xóa khỏi CSDL**

ád

# Bài 22. Quản lý nhập hàng (productstore)

## **1. Hiển thị danh sách danh mục**

**Bước 1. Tổng quan**

* Phương thức: get
* Url: admin/category/index
* Controller: backend/CategoryControlller
* Hàm (function): index

**Bước 2. Khai báo route (routes/web.php)**

use App\Http\Controllers\backend\CategoryController;

Route::get('category/index', [CategoryController::class, 'index']);

**Bước 3. Viết mã lệnh**

public function index()

{

$categorys = Category::where(‘status’, ‘!=’,0)

->orderBy(‘created\_at’,‘desc’)

->get();

return view(‘backend.category.index’, compact(‘categorys’));

}

## **2. Hiển thị danh sách thùng rác loại**

sdf

## **3. Thêm**

sds

## **4. Chi tiết**

adsa

## **5. Cập nhật**

sads

## **6. Thay đổi trạng thái**

áds

## **7. Xóa vào thùng rác**

áds

## **8. Khôi phục rác**

adssd

## **9. Xóa khỏi CSDL**

ád

# Bài 23. Quản lý xuất hàng (order\_orderdetail)

## **1. Hiển thị danh sách danh mục**

**Bước 1. Tổng quan**

* Phương thức: get
* Url: admin/category/index
* Controller: backend/CategoryControlller
* Hàm (function): index

**Bước 2. Khai báo route (routes/web.php)**

use App\Http\Controllers\backend\CategoryController;

Route::get('category/index', [CategoryController::class, 'index']);

**Bước 3. Viết mã lệnh**

public function index()

{

$categorys = Category::where(‘status’, ‘!=’,0)

->orderBy(‘created\_at’,‘desc’)

->get();

return view(‘backend.category.index’, compact(‘categorys’));

}

## **2. Hiển thị danh sách thùng rác loại**

sdf

## **3. Thêm**

sds

## **4. Chi tiết**

adsa

## **5. Cập nhật**

sads

## **6. Thay đổi trạng thái**

áds

## **7. Xóa vào thùng rác**

áds

## **8. Khôi phục rác**

adssd

## **9. Xóa khỏi CSDL**

ád

# Bài 24. Quản lý khuyến mãi(productsale)

## **1. Hiển thị danh sách danh mục**

**Bước 1. Tổng quan**

* Phương thức: get
* Url: admin/category/index
* Controller: backend/CategoryControlller
* Hàm (function): index

**Bước 2. Khai báo route (routes/web.php)**

use App\Http\Controllers\backend\CategoryController;

Route::get('category/index', [CategoryController::class, 'index']);

**Bước 3. Viết mã lệnh**

public function index()

{

$categorys = Category::where(‘status’, ‘!=’,0)

->orderBy(‘created\_at’,‘desc’)

->get();

return view(‘backend.category.index’, compact(‘categorys’));

}

## **2. Hiển thị danh sách thùng rác loại**

sdf

## **3. Thêm**

sds

## **4. Chi tiết**

adsa

## **5. Cập nhật**

sads

## **6. Thay đổi trạng thái**

áds

## **7. Xóa vào thùng rác**

áds

## **8. Khôi phục rác**

adssd

## **9. Xóa khỏi CSDL**

ád

**11. Chức năng trang người**

**11.1. Lấy dữ liệu cho menu**

Chức năng: Lấy dữ liệu hiển thị menu trang người

- Theo vị trí: position: mainmenu, footermenu

- Theo cấp menu: parent\_id=0 là cấp cha, ngược lại là cấp con

Phương thức: Route::get

URL: menu\_list/{position}/{parent\_id?}

Controller: Api/MenuController

Hàm: menu\_list

public function menu\_list($position, $parent\_id = 0)

{

$args = [

['position', '=', $position],

['parent\_id', '=', $parent\_id],

['status', '=', 1]

];

$menus = Menu::where($args)

->orderBy('sort\_order', 'ASC')

->get();

return response()->json(

[

'success' => true,

'message' => 'Tải dữ liệu thành công',

'menus' => $menus

],

200

);

}

**11.2. Lấy dữ liệu cho slider**

Chức năng: Lấy dữ liệu hiển thị menu trang người

- Theo vị trí: position: slidershow, logo, ads

Phương thức: Route::get

URL: slider\_list/{position}

Controller: Api/SliderController

Hàm: slider\_list

public function slider\_list($position)

{

$args = [

['position', '=', $position],

['status', '=', 1]

];

$sliders = Slider::where($args)

->orderBy('sort\_order', 'ASC')

->get();

return response()->json(

[

'success' => true,

'message' => 'Tải dữ liệu thành công',

'sliders' => $sliders

],

200

);

}

**11.3. Lấy dữ liệu của danh mục sản phẩm(category)**

Chức năng: Lấy dữ liệu hiển thị menu trang người

- Theo cấp category: parent\_id=0 là cấp cha, ngược lại là cấp con

Phương thức: Route::get

URL: category\_list/{parent\_id?}

Controller: Api/CategoryController

Hàm: category\_list

public function category\_list($parent\_id = 0)

{

$args = [

['parent\_id', '=', $parent\_id],

['status', '=', 1]

];

$categorys = Category::where($args)

->orderBy('sort\_order', 'ASC')

->get();

return response()->json(

[

'success' => true,

'message' => 'Tải dữ liệu thành công',

'categorys' => $categorys

],

200

);

}

**11.4. Lấy dữ liệu của sản phẩm(product)**

Chức năng: Lấy dữ liệu hiển thị menu trang người

- Lấy nhiều nhất $limit mẫu tin

- Theo mã loại category\_id=1

Phương thức: Route::get

URL: product\_home/{limit}/{category\_id}

Controller: Api/ProductController

Hàm: product\_home

public function product\_home($limit, $category\_id = 0)

{

$listid = array();

array\_push($listid, $category\_id + 0);

$args\_cat1 = [

['parent\_id', '=', $category\_id + 0],

['status', '=', 1]

];

$list\_category1 = Category::where($args\_cat1)->get();

if (count($list\_category1) > 0) {

foreach ($list\_category1 as $row1) {

array\_push($listid, $row1->id);

$args\_cat2 = [

['parent\_id', '=', $row1->id],

['status', '=', 1]

];

$list\_category2 = Category::where($args\_cat2)->get();

if (count($list\_category2) > 0) {

foreach ($list\_category2 as $row2) {

array\_push($listid, $row2->id);

}

}

}

}

$products = Product::where('status', '=', 1)

->whereIn('category\_id', $listid)

->orderBy('created\_at', 'DESC')->limit($limit)->get();

return response()->json(

[

'success' => true,

'message' => 'Tải dữ liệu thành công',

'products' => $products

],

200

);

}

**11.5. Lấy ra bài viết mới nhất**

Lấy ra nhiều nhất 8 bài viết mới nhất

- Limit: số mẫu tin

- Type="post"

Route::get

Url: post\_list/{type}/{limit}

Controller: Api/PostController

Hàm: post\_list

function post\_list($type, $limit)

{

$args = [

['type', '=', $type],

['status', '=', 1]

];

$posts = Post::where($args)

->orderBy('created\_at', 'DESC')

> limit($limit)

->get();

return response()->json(

[

'success' => true,

'message' => 'Tải dữ liệu thành công',

'posts' => $posts

],

200

);

}

**11.6. Lấy ra tất cả sản phẩm có phân trang**

Lấy ra tất cả sản phẩm có phân trang

- Limit: số mẫu tin

- Page: Trang thứ mấy

Route::get

Url: product\_all/{limit}/{page?}

Controller: Api/ProductController

Hàm: product\_all

public function product\_all($limit, $page = 1)

{

$offset = ($page - 1) \* $limit;

$products = Product::where('status', 1)

->orderBy('created\_at', 'DESC')

->offset($offset)

->limit($limit)

->get();

return response()->json(

[

'success' => true,

'message' => 'Tải dữ liệu thành công',

'products' => $products

],

200

);

}

**11.7. Lấy ra tất cả sản phẩm theo loại có phân trang**

Lấy ra nhiều nhất 8 bài viết mới nhất

- Category\_id: mã loại

- Limit: số mẫu tin

- Page: Trang thứ mấy

Route::get

Url: product\_category/{category\_id}/{limit}/{page?}

Controller: Api/ProductController

Hàm: product\_category

public function product\_category($category\_id, $limit, $page = 1)

{

$listid = array();

array\_push($listid, $category\_id + 0);

$args\_cat1 = [

['parent\_id', '=', $category\_id + 0],

['status', '=', 1]

];

$list\_category1 = Category::where($args\_cat1)->get();

if (count($list\_category1) > 0) {

foreach ($list\_category1 as $row1) {

array\_push($listid, $row1->id);

$args\_cat2 = [

['parent\_id', '=', $row1->id],

['status', '=', 1]

];

$list\_category2 = Category::where($args\_cat2)->get();

if (count($list\_category2) > 0) {

foreach ($list\_category2 as $row2) {

array\_push($listid, $row2->id);

}

}

}

}

$offset = ($page - 1) \* $limit;

$products = Product::where('status', 1)

->whereIn('category\_id', $listid)

->orderBy('created\_at', 'DESC')

->offset($offset)

->limit($limit)

->get();

return response()->json(

[

'success' => true,

'message' => 'Tải dữ liệu thành công',

'products' => $products

],

200

);

}

**11.8. Lấy ra tất cả sản phẩm theo thương hiệu có phân trang**

- Brand\_id: mã thương hiệu

- Limit: số mẫu tin

- Page: Trang thứ mấy

Route::get

Url: product\_brand/{brand\_id}/{limit}/{page?}

Controller: Api/ProductController

Hàm: product\_brand

public function product\_brand($product\_id, $limit, $page = 1)

{

$offset = ($page - 1) \* $limit;

$products = Product::where([['brand\_id', '=', $product\_id], ['status', '=', 1]])

->orderBy('created\_at', 'DESC')

->offset($offset)

->limit($limit)

->get();

return response()->json(

[

'success' => true,

'message' => 'Tải dữ liệu thành công',

'products' => $products

],

200

);

}